**6.5240 Sublinear Time Algorithms** November 4, 2024

# Lecture 19

*Lecturer: Ronitt Rubinfeld* *Scribe: Sruthi Parthasarathi*

Today in lecture, we discussed the adjacency matrix model for representing graphs, and how it can be used to test for bipartiteness of dense graphs in sublinear time.

## 1 Adjacency Matrix Model

### 1.1 Definition

Recall that previously, we worked with the adjacency list model, in which we we were given access to a list of neighbors for each node in the input graph $G$. In contrast, the adjacency matrix model represents input graph $G = (V, E)$ as an $n \times n$ matrix $A$, such that $A_{ij} = 1$ if edge $(i, j)$ is in $E$ and 0 otherwise.

### 1.2 Comparing Models

To begin, note that the adjacency list is a natural representation for sparser graphs, which we will call the set of graphs with maximum degree $\Delta$. The size of the representation is $\Sigma_{v \in V} \deg v \leq \Delta * n$, which is smaller than the corresponding adjacency matrix of size $n^2$ when $\Delta$ is much smaller than $n$. For dense graphs, both representations are $O(n^2)$ in size, so we naturally use the adjacency matrix model.

### 1.3 Generalizing Notions of Distance

We can then generalize our previous notion of $\epsilon$-close to a property to mean that a graph requires fewer than $\epsilon$ fraction of its edges to be changed in order to have the desired property. This works out to be fewer than $\epsilon(\Delta n)$ edges for sparse graphs, and suggests a natural analog of fewer than $\epsilon n^2$ edges for dense graphs.

**Definition 1** *For dense $G$, $G$ is $\epsilon - far$ from $P$ if more than $\epsilon n^2$ entries of $A$ must be changed to turn $G$ into a member of $P$.*

### 1.4 Sparse Properties

With the room to edit up to $\epsilon n^2$ edges, property testing becomes trivial for certain "sparse" properties. For example, a graph only requires $n - 1$ edges to be connected, making all graphs $\epsilon - close$ to connected when using the adjacency matrix model.

In addition to representation size, this generalized notion of distance is another reason that we have different models for sparse and dense graphs – a trivial tester on degree-bounded graphs can simply delete all of the edges and deterministically pass or fail (depending on P) on the empty graph.

This new model is therefore most useful for not-sparse properties on not-sparse graphs.

### 1.5 Sidenote: The Source of Hardness

An interesting result is that one can test closeness to 3-colorability (an NP-hard problem) in $O(1)$ time for dense graphs but the same probem requires $\Omega(n)$ queries for sparse graphs. This seems to imply, as one might already intuit, that the hardness of correctly distinguishing graphs in P from those outside arises primarily from graphs that are very close to P.

## 2 Testing Bipartiteness

### 2.1 Definitions

**Definition 2** *A graph $G$ is **bipartite** if its nodes can each be colored either red or blue such that no edge is monochromatic.*

Note that the following is another equivalent definition:

**Definition 3** *A graph $G$ is **bipartite** if its nodes can be partitioned into $(V_1, V_2)$ such that $\nexists e = (u, v) \in E$ such that both $u$ and $v$ are in the same block of the partition.*

We call any such edges with both endpoints in the same block **violating edges**.

**Definition 4** *A graph $G$ is **$\epsilon$-far from bipartite** if we must remove more than $\epsilon n^2$ edges to make it bipartite. Equivalently, for all possible partitions $(V_1, V_2)$, there are at least $\epsilon n^2$ violating edges.*

### 2.2 Some Algorithms

#### 2.2.1 Exact Bipartiteness

This requires $O(n)$ time, and can be accomplished using BFS.

#### 2.2.2 Sparse Graphs

One possible algorithm that was discussed ($\Theta(\sqrt{n})$ complexity) relies on the fact that bipartite graphs do not contain cycles of odd length, and is roughly as follows:

1. Pick a random node $v$
2. Take a random walk of length $\log n$ from $v$
3. Repeat the above two steps multiple times
4. Look for cycles (either by taking the union of all of the random walks or by checking endpoints of the walks to see if any of them are connected
5. If any cycles of odd length exist, FAIL
6. Else, PASS

#### 2.2.3 Dense Graphs

In contrast, similar to 3-colorability, we can actually test for this property in dense graphs in constant time (independent of $n$) using the fact that any subgraph of a bipartite graph is also bipartite – below is one such algorithm.

1. Pick a sample of $O(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$ nodes $S$
2. Look at the induced graph on $S$ (only includes edges between nodes in $S$) – this corresponds to the $|S| \times |S| = O(\frac{1}{\epsilon^4} \log \frac{1}{\epsilon^2})$ size submatrix of the adjacency matrix of $G$
3. Perform BFS (this takes little time because the set of nodes is small) and PASS if bipartite
4. Else, FAIL

## 3 Roadblocks to Proving Correctness

As with all property testing algorithms, proving correctness requires two things:

1. The algorithm passes on bipartite $G$.

2. The algorithm fails on $G$ that are $\epsilon$-far from bipartite.

For the algorithm on dense graphs described in 2.2.3, the first condition follows from the fact that any subgraph of a bipartite graph is also bipartite. Therefore our algorithm will always pass on bipartite $G$. However, we run into some difficulties in our analysis when we take a look at $\epsilon$-far $G$.

### 3.1 Partition Information

Even if we sample a violating edge of a partition $(V_1, V_2)$ of $G$, how can we quickly determine that it is a violating edge? We will discuss this in depth in later sections, where we will introduce partition oracles that are able to determine which part of a partition each of an edge's endpoints lie in.

### 3.2 Query Complexity

A second issue that arises is that even if we sample a violating edge of a partition $(V_1, V_2)$ of $G$ that is bad (has more than $\epsilon n^2$ violating edges), it is possible that our sample still looks bipartite. This is because a BFS will yield a valid partition as long as even one valid partition exists, and no edge is a violating edge from the perspective of all $2^n$ partitions.

Our first instinct might be to simply increase our sample size with the hope that we see at least one violating edge for each and every possible partition. For a sample containing $\Theta(\frac{1}{\epsilon}\log\frac{1}{\delta})$ edges and a given partition $(V_1, V_2)$ of $G$, the probability that none of the sampled edges are violating edges is at most $(1-\epsilon)^{\frac{1}{\epsilon}\log\frac{1}{\delta}} = \delta$. We can then use the union bound to see that the probability every partition avoids having a violating edge sampled is at most the number of partitions times $\delta$, which is $2^n * \delta$. For this to be sufficiently small, we can choose $\delta \ll 2^{-n}$.

But... this is no longer a sublinear query complexity! With this dependence of $\delta$ on $n$, our sample must contain at least $\Theta(\frac{1}{\epsilon}\log\frac{1}{\delta}) = \Theta(\frac{1}{\epsilon}\log 2^n) = \Theta(\frac{n}{\epsilon})$ edges (and therefore the adjacency matrix itself has size at least $\Theta(\frac{n}{\epsilon})$).

## 4 Representative Partitions

We start by addressing the second issue, which motivates us to avoid union bounding over all $2^n$ partitions. Instead, we seek to construct a subset of partitions $R$, where $|R| \ll 2^n$ but the partitions in $R$ are still representative enough of the entire set of partitions $P$ to catch graphs that are $\epsilon$-far from bipartite.

### 4.1 Desired Properties

We begin by enumerating conditions for a useful set of representatives $R$:

1. $R \subseteq P$

2. $|R| \ll |P|$: This reduction in size allows our union bounding to be more fruitful.

3. $\forall p \in P, \exists r \in R$ such that $d(p, r) \leq \epsilon'$: This ensures that the smaller set is still meaningful in some way. Going forwards, we will define $d(p, r)$ such that $d(p, r) \leq \epsilon'$ means that there are fewer than $\epsilon' n^2$ crossing edges in the symmetric difference of the two partitions.

With a set $R$ that satisfies the above properties (sometimes called an $\epsilon$-net) for $\epsilon' = \frac{\epsilon}{2}$, we see that:

1. For bipartite $G$, there exists a bipartition $p \in P$, and therefore a partition $r \in R$ with at most $\frac{\epsilon n^2}{2}$ violating edges as a result of Condition 3.

2. For $G$ that are $\epsilon$-far from bipartite, every $p \in P$ has at least $\epsilon n^2$ violating edges, so every $r \in R$ also has at least $\epsilon n^2$ violating edges as a result of Condition 1.

## 5 Proposed Algorithm

This allows us to design the following algorithm:

1. Pick a set $U$ of $\Theta(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ nodes randomly from $V$.
2. if U is not bipartite, FAIL.
3. Pick a set $U'$ of $\Theta(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ nodes randomly from V, and let $W$ be the set of pairs formed by pairing adjacent nodes in $U'$ .
4. For all possible bipartitions of $U$ into $U_1$ and $U_2$:
   (a) Define an oracle that partitions the entire graph into $Z_1$ and $Z_2$
   (b) $\forall u \in U'$ call the oracle to see if $u$ is in $Z_1$ or $Z_2$
   (c) Count the number of pairs in $W$ that are violating edges of $(Z_1, Z_2)$
   (d) If fewer than $\frac{3}{4}\epsilon|W|$ are violating edges, output PASS
   (e) Else, continue to the next partition
5. FAIL

Some things to note:

- The first two steps are the same as our initial algorithm, which sampled a small subset of nodes and checked for bipartiteness on the induced graph using BFS.
- Our choice of $U$ influences the representative partitions we look at, as each partition of $U$ induces a partition via the oracle in Step 4.
- Our choice of $U'$ serves as a test set by providing a list of pairs that will be checked for violating edges.
- We don't require 0 violations in order to pass because only representative partitions are being tested, and it is possible for a bipartite graph to not have any good bipartitions in the representative set.
- The number of queries is still independent of $n$ because there are only $2^{|U|} = \text{poly}(\frac{1}{\epsilon})$ bipartitions being tested in Step 4.

### 5.1 Partition Oracles

We now elaborate on what the partition oracle from Step 4 of our algorithm looks like. The goal of such an oracle is to take in a partition of $U$ into $U_1, U_2$ and partition the whole graph into $Z_1, Z_2$ such that for any query node $v$, the oracle can output whether $v$ is in $Z_1$ or $Z_2$. The oracle can do this as follows:

Output $Z_1$ if...

- $v \in U_1$
- $v$ has a neighbor in $U_2$ but not in $U_1$

- $v$ has no neighbor in $U$

Else output $Z_2$ if...

- $v \in U_2$
- $v$ has a neighbor in $U_1$ but not in $U_2$

Else output "bad partition". Note that the oracle only gets here if a node has a neighbor in both $U_1$ and $U_2$. In addition, we can see that the source of error for actual bipartite graphs comes from arbitrarily placing all nodes with no neighbors in $U_1$ or $U_2$ into $Z_1$, as there may be edges between pairs of such nodes.

## 5.2 Runtime Analysis

The oracle requires $O(|U|)$ time per query because it requires checking if each element of $U$ is a neighbor of the query node $v$, rather than iterating through all of $v$'s neighbors. The algorithm itself queries the partition oracle for each element of $U'$ for each possible bipartition of $U$, resulting in a runtime of $O(2^{|U|} * |U'|) = O(2^{\frac{1}{\epsilon}\log\frac{1}{\epsilon}} \times \frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$. These are both independent of $n$, as desired!

## 5.3 Proof of Correctness

### 5.3.1 $G$ is $\epsilon$-far

In this case, every partition tested by the algorithm has at least $\epsilon n^2$ violating edges. So for all $Z_1, Z_2$ generated by $U_1, U_2$:

Pr[fraction of violating edges in W $\leq \frac{3}{4}\epsilon] \ll \frac{1}{8 \cdot 2^{|U|}}$ (Chernoff bound)

Pr[algorithm passes $\leq 2^{|U|} \cdot \frac{1}{8 \cdot 2^{|U|}} \ll \frac{1}{8}$ (Union bound)

As intended, the smaller size of the representative set generated by bipartitions of $U$ resulted in a more practical union bound.

### 5.3.2 $G$ is bipartite

We know the first step will not fail, because any $U$ that is chosen will also be bipartite. Now, let $Y_1, Y_2$ be the "true" bipartition of $G$. Since we iterate through all possible bipartitions of $U$, we will also look at the one that is consistent with $Y_1, Y_2$ – let this be $U_1, U_2$ going forward. We wish to show that this bipartition passes.

Now, what happens when we look at this bipartition? We first define a partition oracle that creates $Z_1, Z_2$, and then use it to test each pair of nodes in $W$. The error then, arises from places in which $Z_1$ and $Z_2$ differ from $Y_1$ and $Y_2$. If you recall from the earlier section on partition oracles, this difference is due to cases in which nodes from $Y_2$ are placed arbitrarily into $Z_1$ because they have no edges to $U_1$ or $U_2$, as every other assignment should be consistent with the true bipartition.

So let us take a closer look at those nodes – let $V' \subset V$ be the set of all nodes with no edges to $U$. The number of violating edges we count in Step 4 of our algorithm is equivalent to the number of pairs in $W$ that have both endpoints in $V'$ and correspond to an actual edge in the graph. We can therefore upper bound this by the total number of edges that are incident to any node in $V'$.

In order to do so, we first divide $V'$ into two groups based on node degree. Let $L$ be the set of all $v \in V'$ such that $v$ has no neighbors in $U$ and $\deg(v) < \frac{\epsilon}{4n}$, and $H = V \setminus L$. Since the number of edges between nodes in $V'$ is less than the total number of edges incident to any node in $V'$, we bound that instead:

1. For $v \in L$, $\deg(v) < \frac{\epsilon}{4n}$, so there are at most $|L| * \frac{\epsilon}{4n} \leq n * \frac{\epsilon}{4n} = \frac{\epsilon}{4}$ edges incident to $v \in L$

2. For $v \in H$, we only know that $\deg(v) < n$, but we claim that $|H| < \frac{\epsilon}{4n}$, so there are still at most $|H| * n \leq \frac{\epsilon}{4n} * n = \frac{\epsilon}{4}$ edges incident to $v \in H$

Therefore, there are very few violating edges, as desired! Now, last but not least, why can we assume that $|H| < \frac{\epsilon}{4n}$? Not quite a rigorous answer, but the intuition is that each of the nodes in $H$ is high degree but still didn't happen to be incident to any node in a randomly chosen set ($U$)! This is quite unlikely, so the expected number of such nodes must be quite small.